Statistical Techniques in Robotics (16-831, S21) Lecture #12 (Monday, March 15)

# MAB-ExploreExploit -UCB

*Lecturer: Kris Kitani* *Scribes: Jinkun Cao, Yuda Song*

## 1 Review

In the last lecture, we covered AdaBoost [1], an online boosting algorithm that combines weak learner to obtain PAC guarantee. Here we provide a brief review on PAC learning and adaboost.

### 1.1 PAC Learning

PAC stands for "Probably Approximately Correct", and it is a generic framework to analysis the sample complexity of a learning algorithm in order to achieve good performance with high probability.
Here we first introduce some terminology:

- Denote $\mathcal{D}$ be the dataset.
- Let $N$ be the size of the dataset.
- Let $P(x, y)$ be the distribution where $\mathcal{D}$ is drawn from.
- Let $y = f^*(x)$ be the deterministic distribution where the class labels are determined.
- Let $\mathcal{F}$ be the function space and we can say $f(x; \mathcal{D}) \sim \mathcal{F}|\mathcal{D}$.

The goal of PAC learning is to determine the sample size $N$ required such that

$$\mathbb{E}_{p(x,y)}\left[\mathbb{1}\{f(x|\mathcal{D} \neq y)\}\right] < \epsilon$$
$$\text{with probability } 1 - \delta$$

for any dataset $\mathcal{D}$ drawn from $p$ with size $N$.

### 1.2 Adaboost

Here present the algorithm of adaboost: in each round, the algorithm maintains weights for each training data. The algorithm then inquires a weak learner on the weighted dataset. Then the algorithm adjust the weights according to the error of the current weak learner. At the end, the algorithm returns a combination of the weak learning whose weight is determined by the mistake that it makes. The detailed algorithm is presented in Alg. 1.

**Algorithm 1** Adaboost

**Require:** $\mathbf{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N, \{\mathbf{w}_n^{(0)}\}_{n=1}^N, T$

1: **for** $t = 1, \cdots, T$ **do**
2: $\quad \mathbf{p}^{(t)} = \mathbf{w}^{t-1} / \sum_n \mathbf{w}_n^{t-1}$
3: $\quad h^t = \text{WEAKLEARNER}(\mathbf{D}, \mathbf{p}^{(t)})$
4: $\quad \epsilon^{(t)} = \sum_n p_n^t |h^t(\mathbf{x}_n) - y_n|$
5: $\quad \beta^{(t)} = \epsilon^{(t)}(1 - \epsilon^{(t)})$
6: $\quad \mathbf{w}_n^t = \mathbf{w}_n^{(t-1)} \beta^{1-|h^{(t)}(\mathbf{x}_n^{(t)}) - y_n^{(t)}|} \; \forall n$
7: **end for**
8: $h_F(\mathbf{x}) = \mathbb{1}\{\sum_{t=1}^T (\log(\frac{1}{\beta^{(t)}}) h^{(t)}(\mathbf{x})) \geq \frac{1}{2} \sum_{t=1}^T (\log(\frac{1}{\beta^{(t)}})\}$

### 1.2.1 Error bound of Adaboost

Now we provide the error bound of Adaboost. The proof is in the homework.

**Theorem 1** (Mistake bound of Adaboost)**.** *Let $\epsilon$ be the error made by $h_F$. We have*

$$\epsilon \leq 2^T \prod_{t=1}^T \sqrt{\epsilon_t(1 - \epsilon_t)}.$$

## 2 Multi-Armed Bandits

Recall our settings of Multi-Armed Bandits:

- **one shot feedback**: One action leads to one reward (selecting an action doesn't change the next time step)
- **exhaustive feedback**: state is static, sampled from finite space at each time step
- **evaluative feedback**: receive a sampled reward at each time step

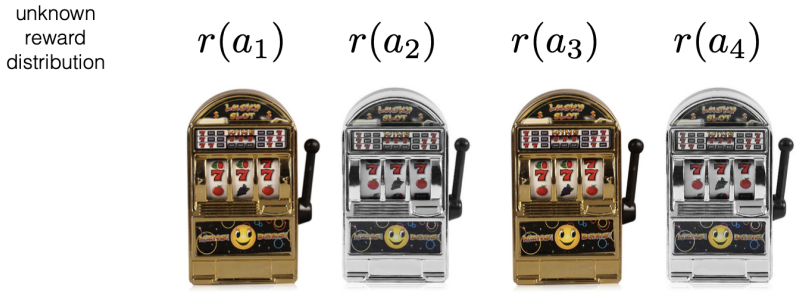


Figure 1: Multi-Armed Bandits.

In this lecture, we will introduce two algorithms: Explore-Exploit and UCB.

| | Context-free | Contextual |
|---|---|---|
| **Stochastic environment** | Explore-Exploit, UCB | linUCB |
| **Adversarial environment** | EXP3 | EXP4 |

Figure 2: Types of MAB algorithms.

# 3 Explore-Exploit

On a very high level, the explore-exploit algorithm contains two phases:

- **Explore Phase:** Pull each arm $M$ times to estimate the mean reward.
- **Exploit Phase:** Next keep pulling the arm with the highest expected mean reward until T.

## 3.1 Notation

Let's first introduce the notation that will be of use:

- $a_k \in \mathcal{A}$: action (arm)
- $|\mathcal{A}| = K$: dimension of the action space
- $r^{(t)}$: reward (received at time t)
- $M$: Number of exploration steps (per action)
- $\widehat{\mu}_k$: Estimated average reward of action $k$

## 3.2 Algorithm

Now we provide the psuedocode of the algorithm in Alg. 2.

**Algorithm 2** Explore-Exploit

**Require:** $M$

**for** $k = 1, \cdots, K$ **do**
  **for** $m = 1, \cdots, M$ **do**
    $a = k$
    Receive($r$)
    $\widehat{\mu}_k = \widehat{\mu}_k + \frac{r}{M}$
  **end for**
**end for**
**for** $t = KM, \cdots, T$ **do**
  $a^{(t)} = \arg\max_k \widehat{\mu}'_k$
**end for**

### 3.3 Regret Analysis

The algorithm, although looks simple, actually enjoys a sub-linear regret. Before we gives the theorem and proofs of the regret bound, let's first introduce one of the most well-known concentration inequality that will be helpful for our analysis.

**Lemma 2** (Hoeffding's Inequality [2])**.** *Consider a one-dimension distribution $\nu$ with expectation mu, where any sample $r \sim \nu$ bounded such that $r \in [0, 1]$. Given $T$ i.i.d. samples $\{r^{(t)}\}_{t=1}^{T}$, we have that for any $\epsilon$,*

$$\mathbb{P}\left(\left|\frac{1}{T}\sum_{t=1}^{T} r^{(t)} - \mu\right| \geq \epsilon\right) \leq 2\exp\left(-2T\epsilon^2\right).$$

Then we can see that if the above inequalities holds, we will have

$$\mathbb{P}\left(\left|\frac{1}{T}\sum_{t=1}^{T} r^{(t)} - \mu\right| < \epsilon\right) \geq 1 - 2\exp\left(-2T\epsilon^2\right).$$

Now if we want the inequality to hold for some high probability, say with probability $1 - \delta$ where $\delta$ is small, we can then plug in $\delta$ into the above inequality and solve for $\epsilon$:

$$\delta = 2\exp\left(-2T\epsilon^2\right).$$

With some algebra we get:

$$\epsilon = \sqrt{\frac{\log(2/\delta)}{2T}}.$$

This threshold, $\epsilon$, is called "confidence interval" or "confidence bound", as illustrated in Fig. 3.

Now we are ready to provide the regret bound of Explore-Exploit algorithm:

**Theorem 3** (Regret Bound for Explore-Exploit)**.** *The regret bound of Explore-Exploit is:*

$$R = \widetilde{\mathcal{O}}(K^{1/3}T^{2/3}).$$

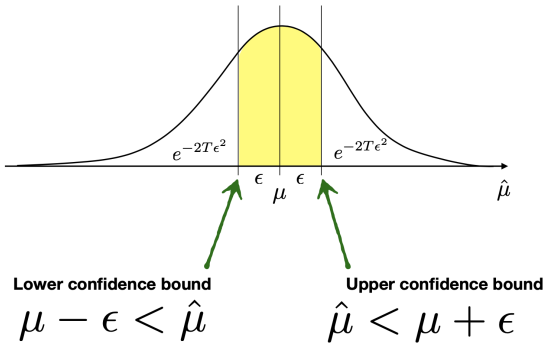


Figure 3: An illustration of confidence interval.

*Proof.* Let's first bound the regret in the exploring phase: suppose in the worst case, each pull gives us 0 reward but the best is 1, then with $KM$ pulls in the exploring phase, we have

$$R_{\text{explore}} = \mathcal{O}(KM).$$

In the exploitation phase, if we pulled the wrong (not the true best) arm, that means our estimate was wrong:

$$\widehat{\mu}_{\widehat{k}} \geq \widehat{\mu}_{k^*},$$

where $\widehat{k} = \arg\max_k \widehat{\mu}_k$ and $k^* = \arg\max_k \mu_k$. And this will serve as our potential function (thus we need to upper bound the LHS and lower bound the RHS and then combine).
To upper bound the LHS, recall that using Hoeffding we get the upper confidence bound of $\widehat{\mu}_{\widehat{k}}$:

$$\widehat{\mu}_{\widehat{k}} \leq \mu_{\widehat{k}} + \epsilon = \mu_{\widehat{k}} + \sqrt{\frac{\log(2/\delta)}{2M}}.$$

So similarly, we can lower bound the RHS by its lower confidence bound:

$$\widehat{\mu}_{k^*} \geq \mu_{k^*} = \mu_{k^*} - \sqrt{\frac{\log(2/\delta)}{2M}}.$$

Combine the above two inequalities we get:

$$\mu_{k^*} - \mu_{\widehat{k}} \leq 2\sqrt{\frac{\log(2/\delta)}{2M}}.$$

Thus we can bound the regret in the exploitation phase:

$$\begin{aligned} R_{\text{exploit}} &= (T - MK)(\mu_{k^*} - \mu_{\widehat{k}}) \\ &\leq (T - MK)2\sqrt{\frac{\log(2/\delta)}{2M}} \\ &\leq \mathcal{O}\left(2(T - MK)\sqrt{\frac{1}{M}}\right). \end{aligned}$$

Since the overall regret is just the sum of the regrets in the two phases, we have:

$$\begin{aligned} R &= R_{\text{explore}} + R_{\text{exploit}} \\ &\leq \mathcal{O}(MK) + \mathcal{O}\left(2(T - MK)\sqrt{\frac{1}{M}}\right) \\ &\leq \mathcal{O}(K^{1/3}T^{2/3}), \end{aligned}$$

where in the second inequality we take $M = \left(\frac{T}{K}\right)^{2/3}$. □

Now we can see that Explore-Exploit is a no regret algorithm!

# 4 UCB

Now we introduce the UCB algorithm. It is similar to Explore-Exploit algorithm in the sense that it also take use of the upper confidence bound, but here we skip the explore phase and directly use the upper confidence bound as a bonus at each timestep.

## 4.1 Notation

As usual, let's first introduce the notation:

- $k^{(t)}$: the arm pulled at time $t$
- $r^{(t)}$: reward received at time $t$
- $T_k^{(t)}$: number of times arm $k$ was pull up to time $t$
- $\mu_k$: true mean reward of arm $k$
- $\widehat{\mu}_k^{(t)}$: estimated mean reward of arm $k$ at time $t$
- $T$ time horizon

## 4.2 Algorithm

Now we provide the algorithm in Alg. 3.

**Algorithm 3** UCB

**Require:** $\delta, T$

**for** $t = 1, \cdots, T$ **do**
  **if** $t \leq k$ **then**
    $k = t$
  **else**
    $k = \arg\max_{k'} \left( \widehat{\mu}_{k'} + \sqrt{\frac{\log(2T/\delta)}{2T_{k'}^{(t-1)}}} \right)$
  **end if**
  Receive($r^{(t)}$)
  $T_k^t = T_k^{(t-1)} + 1$
  $T_{k'}^t = T_{k'}^{(t-1)}\ \forall k' \in [K] \setminus \{k\}$
  $\widehat{\mu}_k = \frac{1}{T_k^{(t)}} \left( T_k^{(t-1)} \widehat{\mu}_k + r_k^{(t)} \right)$
**end for**

### 4.3 Regret Analysis

Before we give out the regret bound on UCB, we first introduce the following lemma:

**Lemma 4** (Union Bound)**.** *Let $x_i$ be a set of random variables. We have*

$$\mathbb{P}\left( \bigcup_i x_i \right) \leq \sum_i \mathbb{P}(x_i).$$

But how is this lemma going to help? With this lemma, now we can bound the probability of events for every round. I.e.,

$$\begin{aligned} \mathbb{P}\left( \bigcup_{t=1}^{T} |\widehat{\mu}_k^{(t)} - \mu_k| > \sqrt{\frac{\log(2/\delta)}{2T_k^{(t)}}} \right) &\leq \sum_{t=1}^{T} \mathbb{P}\left( |\widehat{\mu}_k^{(t)} - \mu_k| > \sqrt{\frac{\log(2/\delta)}{2T_k^{(t)}}} \right) \\ &\leq T\delta. \end{aligned}$$

Then let $\delta' \triangleq T\delta$ we get

$$\epsilon' = \sqrt{\frac{\log(2T/\delta')}{2T_k^{(t)}}}. \tag{1}$$

With these knowledge equipped, we are ready to present the regret bound for UCB algorithm:

**Theorem 5** (Regret bound for UCB)**.** *The regret bound for UCB is:*

$$R = \mathcal{O}\left( 2\sqrt{\log(2T/\delta)\sqrt{KT}} \right) = \widetilde{\mathcal{O}}(\sqrt{KT}).$$

*Proof.* Similar to the exploit phase in Explore-Exploit, note that if we make a mistake (pulling the non-optimal arm) in round $t$, we have:

$$\widehat{\mu}_k + \sqrt{\frac{\log(2T/\delta)}{2T_k^{(t)}}} \geq \widehat{\mu}_{k^*} + \sqrt{\frac{\log(2T/\delta)}{2T_{k^*}^{(t)}}}.$$

Again let upper bound the LHS and lower bound the RHS with what we get in Eq. 1.
For the LHS:

$$\widehat{\mu}_k + \sqrt{\frac{\log(2T/\delta)}{2T_k^{(t)}}} \leq \mu_k + 2\sqrt{\frac{\log(2T/\delta)}{2T_k^{(t)}}}.$$

For the RHS:

$$\begin{aligned}\widehat{\mu}_{k^*} + \sqrt{\frac{\log(2T/\delta)}{2T_{k^*}^{(t)}}} &\geq \left(\mu_{k^*} - \sqrt{\frac{\log(2T/\delta)}{2T_{k^*}^{(t)}}}\right) + \sqrt{\frac{\log(2T/\delta)}{2T_{k^*}^{(t)}}} \\ &= \mu_{k^*}.\end{aligned}$$

Combine we have

$$\mu_k + 2\sqrt{\frac{\log(2T/\delta)}{2T_k^{(t)}}} \geq \mu_{k^*}.$$

Finally we have:

$$\begin{aligned}R &= \sum_{t=1}^{T}(\mu_{k^*} - \mu_k) \\ &\leq \sum_{t=1}^{T} 2\sqrt{\frac{\log(2T/\delta)}{2T_k^{(t)}}} \\ &= \frac{1}{2}\sqrt{\log(2T/\delta)}\sum_{t=1}^{T}\sqrt{\frac{1}{T_k^{(t)}}} \\ &= \frac{1}{2}\sqrt{\log(2T/\delta)}\sum_{t=1}^{T}\sum_{j=1}^{K}\mathbb{1}\{k^{(t)}=j\}\sqrt{\frac{1}{T_j^{(t)}}} \\ &= \frac{1}{2}\sqrt{\log(2T/\delta)}\sum_{j=1}^{K}\sum_{t=1}^{T}\mathbb{1}\{k^{(t)}=j\}\sqrt{\frac{1}{T_j^{(t)}}} \\ &= \frac{1}{2}\sqrt{\log(2T/\delta)}\sum_{j=1}^{K}\sum_{t=1}^{T_j^{(T)}}\sqrt{\frac{1}{t}} \\ &\leq \sqrt{\log(2T/\delta)}\sum_{j=1}^{K}\sqrt{T_j^{(T)}} \\ &\leq \sqrt{\log(2T/\delta)}K\sqrt{\frac{1}{K}\sum_{j=1}^{K}T_j^{(T)}} \\ &= \sqrt{\log(2T/\delta)}K\sqrt{\frac{T}{K}} \\ &= \sqrt{\log(2T/\delta)}\sqrt{KT} \\ &= \widetilde{\mathcal{O}}(\sqrt{KT}).\end{aligned}$$

Here the second inequality is by $\sum_{t=1}^{T}\sqrt{\frac{1}{t}} \leq 2\sqrt{T}$ and the third inequality is Jensen's inequality. □

## References

[1] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[2] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.